EXHIBIT 20

Filed Under Seal

QUINN EMANUEL URQUHART & SULLIVAN, LLP
Charles K. Verhoeven (Bar No. 170151)
charlesverhoeven@quinnemanuel.com
Melissa Baily (Bar No. 237649)
melissabaily@quinnemanuel.com
Lindsay Cooper (Bar No. 287125)
lindsaycooper@quinnemanuel.com
50 California Street, 22nd Floor
San Francisco, California 94111-4788
Telephone: (415) 875-6600
Facsimile: (415) 875-6700

Attorneys for GOOGLE LLC

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

GOOGLE LLC, Plaintiff v. SONOS, INC., Defendant.	Case No. 3:20-cv-06754-WHA

GOOGLE LLC'S THIRD SUPPLEMENTAL OBJECTIONS AND RESPONSES TO PLAINTIFF SONOS, INC.'S FIRST SET OF FACT DISCOVERY INTERROGATORIES (NO. 15)

Pursuant to Rule 33 of the Federal Rules of Civil Procedure, Defendant Google LLC ("Google") hereby objects and responds to Plaintiff Sonos, Inc.'s ("Sonos") First Set of Fact Discovery Interrogatories to Defendant ("Interrogatories"). Google responds to these Interrogatories based on its current understanding and the information reasonably available to Google at the present time. Google reserves the right to supplement these responses if and when additional information becomes available.

OBJECTIONS AND RESPONSES TO FACT DISCOVERY INTERROGATORIES

INTERROGATORY NO. 15:

For each of the YouTube, YouTube Music, YouTube TV, Google Play Music, Google Podcasts, and Spotify media services, describe in detail how an Accused Google Product (e.g., an Accused Cast-Enabled Media Player or Accused Pixel Device) receives and then plays back a sequence of media items (e.g., songs, podcast episodes, etc.) in connection with a given one of the aforementioned media services including, but not limited to, (i) describing in detail any communications between the Accused Google Product and any web server (e.g., Accused Google Server or third-party server) and how such communications take place, (ii) describing in detail how any Accused Google Server generates, maintains, and/or updates a set of one or more media-item "recommendations"[1] that are sent to the Accused Google Product and how those "recommendations" are sent to the Accused Google Product, and (iii) describing in detail how any Accused Google Server facilitates Google's "Autoplay feature"[2] for playback at the Accused Google Product and how the "Autoplay feature" is utilized at the Accused Google Product.

OBJECTIONS: Google incorporates by reference all of its General Objections as if fully set forth herein. Google objects to the characterization of this interrogatory as a single interrogatory given that it contains multiple discrete subparts under Fed. R. Civ. P. 33(a)(1). Google objects to this interrogatory on the grounds that it is vague, ambiguous, unclear as to information sought, and lacking sufficient particularity to permit Google to reasonably prepare a response with respect to the undefined terms "receives and then plays back a sequence of media items," "in connection with a given one of the aforementioned media services," "communications between the Accused Google Product and any web server," "how such communications take place," "generates, maintains and/or updates," "how those 'recommendations' are sent to the Accused Google Product," "how any

[1] *See, e.g.*, https://www.youtube.com/howyoutubeworks/product-features/recommendations/; https://support.google.com/youtubemusic/answer/6313542?hl=en; https://support.google.com/websearch/answer/10017274?hl=en&co=GENIE.Platform%3DAndroid#zippy=.

[2] *See, e.g.*, https://support.google.com/youtube/answer/6327615?hl=en&co=GENIE.Platform%3DAndroid; https://support.google.com/websearch/answer/10017274?hl=en&co=GENIE.Platform%3DAndroid#zippy=.

Accused Google Server facilitates Google's 'Autoplay feature' for playback at the Accused Product," and "how the 'Autoplay feature' is utilized at the Accused Google Product." Google further objects to this interrogatory to the extent that it assumes the existence of hypothetical facts that are incorrect or unknown to Google.

Google also objects to this interrogatory as overbroad, burdensome, and not proportional to the needs of the case, including to the extent it seeks information that is not relevant to any claim or defense of any party or to the subject matter of this action, including to the extent that it seeks information regarding non-accused instrumentalities or technology such as "Spotify media services" and "third-party server[s]." Google further objects to this interrogatory as overbroad and unduly burdensome to the extent that it seeks information that is publicly available, not uniquely within the control of Google, or is equally available to Sonos. Google additionally objects to this interrogatory to the extent it seeks communications and information protected from disclosure by the attorney-client privilege and/or attorney work product doctrine. Google further objects to this interrogatory to the extent it seeks confidential and/or proprietary business information. Google also objects to this interrogatory to the extent that it premature seeks expert discovery, opinion, and/or testimony. Google additionally objects to this interrogatory to the extent it seeks information that is not reasonably accessible or that is not within Google's possession, custody, or control. Google further objects to this interrogatory to the extent it seeks information that is unnecessarily cumulative or duplicative of information sought by other discovery, including Request for Production No. 20.

RESPONSE:

Subject to and without waiving the foregoing General and Specific objections, Google responds, as follows:

Pursuant to Rule 33(d) of the Federal Rules of Civil Procedure, Google further refers Sonos to the source code that Google has made available and the following documents containing information responsive to this interrogatory: GOOG-SONOSWDTX-00005033-53611.

SUPPLEMENTAL RESPONSE: Google maintains the General and Specific objections set forth above. Google further objects to this interrogatory on the grounds that it is vague and

ambiguous to the extent it seeks information regarding products not specifically identified by make or model number in Sonos's infringement contentions. Google also objects to this interrogatory to the extent it seeks to encompass Spotify, which is a separate, third-party application. Subject to and without waiving the foregoing General and Specific objections, Google responds, as follows:

Google refers Sonos to the source code that Google has made available, which is the best evidence of how the devices operate with respect to the operation of the accused functionalities. Pursuant to Rule 33(d) of the Federal Rules of Civil Procedure, Google further refers Sonos to the following documents created during product development which may contain information response to this interrogatory: GOOG-SONOSWDTX-00041650, GOOG-SONOSWDTX-00039521, GOOG-SONOSWDTX-00041722, GOOG-SONOSWDTX-00040397, GOOG-SONOSWDTX-00042266, GOOG-SONOSWDTX-00042272, GOOG-SONOSWDTX-00042282, GOOG-SONOSWDTX-00042365, GOOG-SONOSWDTX-00042378, GOOG-SONOSWDTX-00042380, GOOG-SONOSWDTX-00042385, GOOG-SONOSWDTX-00042397, GOOG-SONOSWDTX-00042402, GOOG-SONOSWDTX-00042404, GOOG-SONOSWDTX-00042413, GOOG-SONOSWDTX-00042754, GOOG-SONOSWDTX-00042954, GOOG-SONOSWDTX-00043052, GOOG-SONOSWDTX-00043318, GOOG-SONOSWDTX-00043323, GOOG-SONOSWDTX-00043799-803, GOOG-SONOSWDTX-00043820, GOOG-SONOSWDTX-00051820, GOOG-SONOSWDTX-00051848, GOOG-SONOSWDTX-00051918, GOOG-SONOSWDTX-00051924, GOOG-SONOSWDTX-00051927, GOOG-SONOSWDTX-00052944-71, GOOG-SONOSWDTX-00051608, GOOG-SONOSWDTX-00051943, GOOG-SONOSWDTX-00037978, GOOG-SONOSWDTX-00051947, GOOG-SONOSWDTX-00037634, GOOG-SONOSWDTX-00053379, GOOG-SONOSWDTX-00036998, GOOG-SONOSWDTX-00037178, GOOG-SONOSWDTX-00037042, GOOG-SONOSWDTX-00037081, GOOG-SONOSWDTX-00037220, GOOG-SONOSWDTX-00040331-83, GOOG-SONOSWDTX-00043467, GOOG-SONOSWDTX-00043471, GOOG-SONOSWDTX-00043550, GOOG-SONOSWDTX-00037146, GOOG-SONOSWDTX-00043548, GOOG-SONOSWDTX-00043676, GOOG-

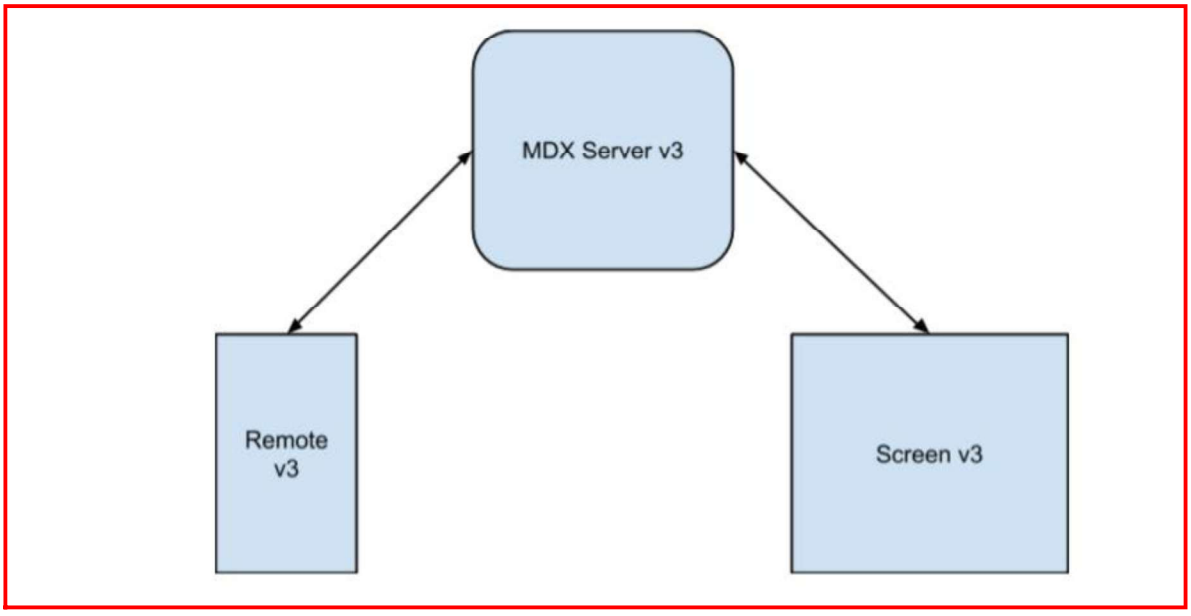
SONOSWDTX-00037178; *see also* GOOG-SONOSWDTX-00005033-8471, GOOG-SONOSWDTX-00022175-371, GOOG-SONOSWDTX-00036346-53611.

SECOND SUPPLEMENTAL RESPONSE:

Google maintains the General and Specific objections set forth above. Google further objects to this interrogatory on the grounds that it is vague, ambiguous, and overbroad to the extent it seeks information regarding products not specifically identified by make or model number in Sonos's infringement contentions, or that have now been dropped by Sonos (*e.g.*, Podcast). Google also objects to this interrogatory to the extent it seeks to encompass Spotify, which is a separate, third-party application. Subject to and without waiving the foregoing General and Specific objections, Google responds, as follows:

Google fully incorporates herein by reference its responses to Interrogatory No. 14. Google further responds that casting operations involving the identified "autoplay" feature are generally described at least in GOOG-SONOSWDTX-00043467, GOOG-SONOSWDTX-00041491 and implemented in the source code provided by Google.

The illustration below provides a simplified view of the general architecture that is employed by devices running MDx applications (e.g., YouTube or YouTube Music):



GOOG-SONOSWDTX-00041650. In this diagram, "Remote v3" represents a Remote client device, such as a phone that runs a YouTube client implementing version 3 of the MDx protocol.

The Remote communicates with the MDx server, which in turn communicates with the Screen. The function in the file cast_mdx_session_service.ts connects the Screen to the MDx server. Once connected, the Screen can receive "methods" (*i.e.*, messages or commands) from the MDx server. An example of one such method is the "setPlaylist" method.

A setPlaylist message is sent from the Remote to the MDx server and relayed to the Screen and requests that the Screen start playing the media identified by the videoId. *Id.* at 8. The parameters of a setPlaylist message sent to the Screen are shown below:

Parameters:

Name	Example	Description
videoId	n_yx_BrdRF8	Video that should start playback ASAP.
currentTime	12.3	Playback start time of **videoId**.
currentIndex	2	The 0-based index of the video in the given list.
vvt	7SlILk2wMTI	**Optional.** The Video Verification Token to make **videoId** playable to the screen.
listId	PLHnyfMqiRRG1u-2MsSQLbXA	List ID that this video is part of. If prefixed with **RQ**, the video is being played from the remote queue.
cat	dYw4meRWGd4	**Optional.** The Content Verification Token to make listId accessible to the screen.
watchNextToken	Us-TVg40ExM_1aH3J8AADPQ	**Optional.** Opaque-to-the-client parameters used by the WatchNextService (usually a seed for the shuffle and/or mix playback).

Id.

In earlier versions of the MDx protocol (Version 2 and earlier) the setPlaylist message sent from the server to the Screen was a "list" of videoIds separated by a comma to represent the current

playlist. In Version 3 of the MDx protocol (released in 2014) the setPlaylist message contains a single videoId (i.e., it is no longer a list) used to identify the video to play. *Id.*

Upon receiving a setPlaylist message, the function handleMessage() creates a PlaybackParams object to be passed to the accused setPlaylist() function. The PlaybackParams class comprises a number of fields:

- eventDetails: a MdxRemoteQueueEvent object
- videoId: string from the videoId field in the incoming "setPlaylist" method
- listId: string from the listId field in the incoming "setPlaylist" method; ID of the remote playlist
- videoCtt: string from the ctt field in the incoming "setPlaylist" method
- listCtt: string from the listCtt field in the incoming "setPlaylist" method
- clickTrackingParams: string derived from the clickTrackingParams field in the incoming "setPlaylist" method
- currentIndex: number derived from the currentIndex field in the incoming "setPlaylist" method
- currentTime: number
- playerParams: string from the playerParams field in the incoming "setPlaylist" method
- watchNextParams: string from the params field in the incoming "setPlaylist" method
- isFling: boolean
- isMdxPlayback: boolean
- isVoiceRequest: boolean
- enableSafetyMode: boolean
- forceReloadPlayback: boolean derived from the forceReloadPlayback field in the incoming "setPlaylist" method

As can be seen, the field "videoId" of the PlaybackParams class is a string from the videoId field in the incoming "setPlaylist" message. To playback a video, the function setPlaylist() uses the PlaybackParams object to play the video specified in the PlaybackParams.videoId field.

An example of a "method" the Screen can receive from the YouTube Frontend service is the "GetWatchNext" method. Generally, at some point after the Screen begins playback of the current media item, GetWatchNext is called by the Screen to request the next WatchEndpPoint. In response to GetWatchNext, the Screen receives WatchNextResponse, a protobuf message, which contains the videoId for the next item to play, along with metadata for the item currently playing, and other elements, such as an autoplay setting, for example.

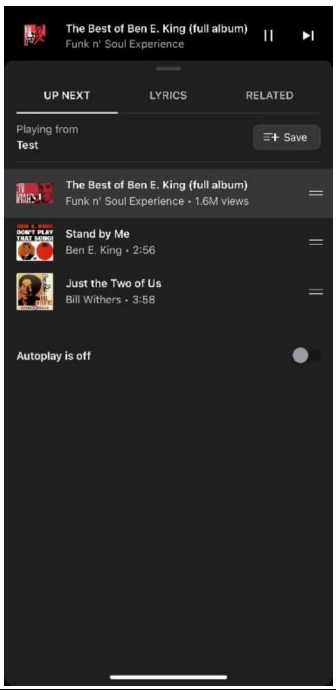
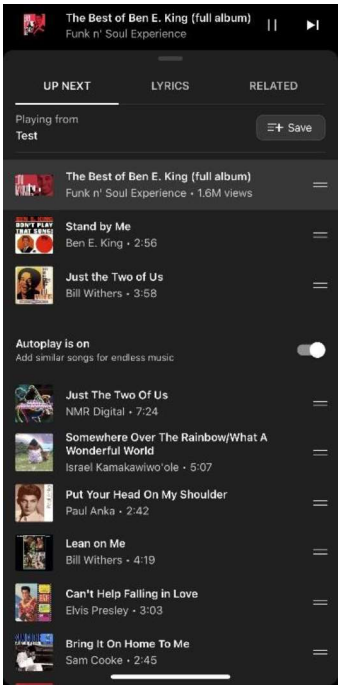
Within the YT Main app, if the user has reached the end of a playlist, or there is no other item to play next, the autoplay feature could be used to continue playing content. Specifically, when

WatchNextResponse is loaded, the local variable upNextVideoId would be used to notify the Remote of what the next autoplay media item is, but upNextVideoId would not be used by any function to perform playback. upNextVideoId is not used for YT Music, at least because YT Music does not enable autoplay.

THIRD SUPPLEMENTAL RESPONSE: Google maintains the General and Specific objections set forth above. Google further objects to this Interrogatory as compound, irrelevant, and unduly burdensome in that it seeks a response "[f]or each of the YouTube, YouTube Music, YouTube TV, Google Play Music, Google Podcasts, and Spotify Media services." For example, Sonos has dropped the "Google Podcasts" and "Spotify" applications from the case. Further, Sonos accused the Google Play Music application of infringing only its '615 patent, and the Court has now granted summary judgment that the "Google Play Music" application does not infringe the '615 patent. Accordingly, Google will limit its response to the "You Tube Music," "YouTube Music," and YouTube TV applications. Google further objects to this interrogatory as vague, ambiguous, overbroad and unduly burdensome, including as to the phrase "describe in detail how an Accused Google Product (e.g., an Accused Cast-Enabled Media Player or Accused Pixel Device) receives and then plays back a sequence of media items (e.g., songs, podcast episodes, etc.) in connection with a given one of the aforementioned media services." The accused YouTube applications and systems have numerous functionalities and features, involve many different playback scenarios, and are comprised of millions of lines of source code. Google also objects to this interrogatory to the extent Sonos is seeking to shift the burden of proving infringement on to Google. It is Sonos's burden, not Google's to prove infringement. Moreover, Sonos's definition of "Accused Google Product" purports to encompass thousands of different (and unspecified) hardware devices. It is not practical for Google to describe "in detail" every playback scenario for every hardware device. Sonos's infringement contentions are also vague and ambiguous as to the particular functionality that it is accusing. Google will provide a reasonable level of detail regarding how the accused applications playback media based on its current understanding of Sonos's contentions.

A Pixel Device running an accused YouTube application allows users to add media items to a playlist on their Pixel Device. For example, the images below are of a YouTube Music application

playing a playlist containing three songs ("The Best of Ben E. King", "Stand by Me", and "Just the Two of Us") on the Pixel Device. Image 1 shows the currently playing song. Images 2 and 3 show a toggle button that can be used by the user to enable or disable "Autoplay." In Image 2, Autoplay is turned off. In Image 3, Autoplay is turned on and additional recommended media items (*e.g.*, "Just the Two of Us," "Somewhere Over the Rainbow," etc.) are displayed below the toggle button. When the user created playlist is exhausted, the Pixel Device will begin playing Autoplay items.

Image 1	Image 2	Image 3
		

Sticking with the above example, the file PlaybackQueue.java specifies the queue interface and the file "DefaultPlaybackQueue.java" implements the queue. They maintain locally, on the Pixel Device, a user-editable list of videos ("QUEUE_LIST") and an Autoplay List with recommended videos ("AUTONAV_LIST"). The user-editable list initially contains information for the three songs ("The Best of Ben E. King", "Stand by Me", and "Just the Two of Us") in the above example. The Autoplay List contains information for the Autoplay items that follow (e.g., "Just the Two of Us," "Somewhere Over the Rainbow," etc.). The user-editable and Autoplay lists are stored locally on the Pixel Device as Android SparseArrays. After the Pixel Device exhausts playback of the user editable list, the first media item in the Autoplay list is added to the user editable list and selected for playback. A user may also select an Autoplay List item at any time, in which case the selected

Autoplay List item (and all preceding items in the Autoplay List) are moved to the QUEUE_LIST and the selected item will be played back. For instance, if the user selects "Somewhere Over the Rainbow," then that item and the previous item in the Autoplay List (i.e., "Just the Two of Us") are added to the "QUEUE_LIST" on the Pixel Device and "Somewhere Over the Rainbow" will be played back.

The Pixel Device communicates with a WatchNext service over the Internet when playing media items on the Pixel Device. *See* PlayerRequestManager.java, WatchNextFetcher.java; *see also* GOOG-SONOSWDTX-00039785. *See* PlayerRequestManager.java[3], WatchNextFetcher.java[4]; *see also* GOOG-SONOSWDTX-00039785.[5] For instance, the Pixel Device transmits a WatchNext request message to the WatchNext service to obtain metadata for the currently playing media item. When starting playback of a playlist, a WatchNext request message may also be used to obtain recommended Autoplay videos. There is not a "remote playback queue" from which these recommended videos are retrieved. Instead, the recommended videos are dynamically generated by a "MixService"—based on criteria such as the video the user is watching, a user's watch history, and various other criteria and algorithm inputs—and returned to the Pixel Device in the WatchNext Response. Recommended videos are only temporarily maintained in the cloud and discarded when, for instance, a user restarts playback of the current playlist.

The Pixel Device also communicates with a Player service over the Internet when playing media items on the device. For example, in addition to the WatchNext request the Pixel Device sends a GetPlayer request to the Player Service. The Player service determines in real-time which Bandaid server in Google's Content Delivery Network a user should talk to for a particular request

---

[3] /java/com/google/android/libraries/youtube/player/net

[4] java/com/google/android/libraries/youtube/player/net

[5] Unlike the situation where a receiver device is playing back media during casting—which may involve the receiver device obtaining a WatchNext Response that includes Autoplay sets with video renderers for a previous (previous_video_renderer) or next (next_video_renderer) command—playback locally on the Pixel Device does not use previous and next video renderers.

and returns to the Pixel Device Bandaid URLs that point to Bandaid servers. The Bandaid URLs are used by Pixel Devices to request chunks of media that are played back. GOOG-SONOSNDCA-00115814 (Edge Streaming and Bandaid CDN) at slide 9; GOOG-SONOSNDCA-00115893 (Life of a Video Request) at 3 ("To serve, ustreamer must translate a player's byte range request into the corresponding 2MB chunk(s), or players may request specific segments (Live, OTF)."). The Bandaid Content Delivery Network will stream back chunks of media to the Pixel Device. *See also* 7-27-2022 Bhattacharjee Rebuttal Rpt., ¶¶94-100.

A Pixel Device may also cast music for playback on a receiver device. When casting, the accused receiver device will playback a cloud queue. Version 3 of Google's "Multi-Device Experience" (or MDx) protocol is used to manage playback on a receiver device during casting. In particular, the YouTube application communicates with the MDx server, which in turn communicates with the receiver device. The function in the file cast_mdx_session_service.ts connects the receiver to the MDx server. Once connected, the receiver device can receive "methods" (*i.e.*, messages or commands) from the MDx server. An example of one such method is the "setPlaylist" method. A setPlaylist message is sent from the Pixel Device running the YouTube application to the MDx server. When the MDx server receives the setPlaylist message from the YouTube application the handleMessage method is invoked (RealLoungeSessionManager[6], line 740). The MDx server then sends a setPlaylist message to the receiver device (RealLoungeSessionManager.java, line 1496. The MDx server then generates another setPlaylist message that its sends to the receiver device. *Id.* at 8. The parameters of a setPlaylist message sent from the MDx server to the receiver device are described above in Google's Second Supplemental Response and include the media item that should begin playing on the receiver device. *See also* 7-27-2022 Bhattacharjee Rebuttal Rpt., ¶88. The receiver device receives the setPlaylist message sent

[6] 2021-02-01_YTServerMDx09292020/google3/java/com/google/youtube/lounge/browserchannel

by the MDx server. In particular, the method handleMessage (loungeadapter.ts[7], line 905) processes the setPlaylist message (loungeadapter.ts, line 924).

After receiving a setPlaylist message, the receiver device sends a WatchNext request message to YouTube's InnerTube service and a GetPlayer request message to YouTube's Player service. *See, e.g,* GOOG-SONOSWDTX-00039491. The WatchNext request message includes, among other things, the videoID of the media currently playing (innertube_watch_next_service.proto, line 97) and the playlistID corresponding to the cloud queue at the MDx server (innertube_watch_next_service.proto, line 103). The WatchNext request is received by the InnerTube servers which invokes handle_get_watch_next (innertube_watch_next.py, line 349). The handle_get_watch_next function invokes _get_watch_next() (innertube_watch_next.py, line 404), which processes the request (innertube_watch_next.py, line 671) and returns to the playback device a WatchNext response (innertube_watch_service.proto; content.py, starting at line 796) with a large volume of information, primarily relating to metadata for the currently playing media item. Within this large volume of information is the videoId for the next media item in the cloud queue. The GetPlayer request causes the Player service to return to the receiver device Bandaid URLs that point to the Bandaid server from which the receiver device should request chunks of the current media item. The receiver will send a get WatchNext request and GetPlayer request each time a new song or video starts playing. *See also* 7-27-2022 Bhattacharjee Rebuttal Rpt., ¶¶90-92.

[7] 2021-02-01_YTReceivers09292020/google3/video/youtube/src/web/javascript/library/mdx/screen_ts

DATED: November 3, 2022

QUINN EMANUEL URQUHART & SULLIVAN, LLP

By: */s/ Charles K. Verhoeven*

Charles K. Verhoeven (*pro hac vice*)
charlesverhoeven@quinnemanuel.com
Melissa Baily (*pro hac vice*)
melissabaily@quinnemanuel.com
Lindsay Cooper (*pro hac vice*)
lindsaycooper@quinnemanuel.com
QUINN EMANUEL URQUHART & SULLIVAN LLP
50 California Street, 22nd Floor
San Francisco, California 94111-4788
Telephone: (415) 875 6600
Facsimile: (415) 875 6700

Counsel for Defendant Google LLC

CERTIFICATE OF SERVICE

The undersigned hereby certifies that all counsel of record who have consented to electronic service are being served with a copy of this document via email on November 3, 2022.

/s/ Nima Hefazi
Nima Hefazi